UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/801,150	03/07/2001	Thierry Bedos	10984.5US01	4307

25883 7590 04/14/2006

HOWISON & ARNOTT, L.L.P
P.O. BOX 741715
DALLAS, TX 75374-1715

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 04/14/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/801,150	BEDOS ET AL.	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 January 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-3,7-20,22,25-40 and 42* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,7-20,22,25-40 and 42* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on January 26, 2006 has been entered. Claims 1-3, 7-20, 22, 25-40 and 42 are pending.

Response to Arguments

2. Applicant's arguments with respect to the claims have been considered but are moot in view of the new ground(s) of rejection, as set forth below. Applicant's amendment necessitated the new ground(s) of rejection.

3. Applicant's argument that Hawkins does not determine the priority order responsive to an order with which the non-core services registered with the core service (remarks, page 8, second paragraph) has been fully considered but it is not persuasive.

Specifically, Applicant characterizes Hawkins as teaching a pre-established priority order (remarks, page 8, second paragraph). Indeed, the priority order in Hawkins represents the order with which non-core services register with the core service using "load points" that are inserted into the code (see, for example, column 5, lines 57-67). Hawkins discloses an example to illustrate how a non-core service is registered with the core service using such a load point (see, for example, column 6, lines 15-19).

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-3, 7, 8, 16-20, 22, 32 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,536,035 to Hawkins (art of record, "Hawkins") in view of U.S. Patent No. 6,807,559 to Budhiraja (art of record, "Budhiraja") in view of U.S. Patent No. 6,311,221 to Raz et al. (now made of record, "Raz").

With respect to claim 1 (currently amended), Hawkins discloses a software engine for application loading a software application onto a user's machine (see, for example, the title), wherein a core service of the application is loaded along with the software engine onto the user's machine to enable the application to commence to operate on the user's machine (see, for example, FIG. 1 and column 4, lines 10-12, which shows loading core classes or services to commence the operation of a program, and column 3, lines 22-26, which shows a software engine that is also loaded), the engine subsequently loading non-core services of the application according to a priority order determined by the engine (see, for example, column 3, lines 22-26, which shows that the engine preloads class files of non-core services during program execution, and column 5, lines 57-67, which shows that the class files of non-core services are loaded in the correct order using a time-based priority order), wherein a non-core service is responsible for

providing a functionality of the application (see, for example, column 4, lines 23-39, which shows that the non-core services provide functionality).

Although Hawkins discloses that the engine determines the priority order for loading the non-core services at run time responsive to a user interaction during execution of the software application (see, for example, column 3, lines 57-65, which shows analyzing user interaction with the program at run-time, and column 4, lines 10-39, which shows determining the order in which classes or services are required for providing functionality corresponding to user interaction) and to an order with which the non-core services registered with the core service (see, for example, column 3, lines 65-67, which shows that the non-core services are registered with the core service in order using load points), Hawkins does not expressly disclose the limitation wherein the engine uniquely determines the priority order for loading the non-core services at run time responsive to a user interaction during each execution of the software application.

However, Budhiraja discloses an analogous software engine for application loading a software application onto a user's machine (see, for example, the abstract). A core service of the application is loaded onto the user's machine (see, for example, column 9, lines 16-27, which shows loading a core applet) to enable the application to commence to operate on the user's machine (see, for example, column 9, lines 27-32), and non-core services are subsequently loaded (see, for example, column 9, lines 32-35, which shows component applets). Budhiraja discloses that the component manager (i.e., the engine) uniquely determines which component applets (i.e., non-core services) to load at run time responsive to a user interaction during each execution of the software application (see, for example, column 9, lines 35-47). This enables

individual users of the software application to load, as needed, components that are not necessarily used by every user (see, for example, column 9, lines 7-13).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the engine of Hawkins with the features taught by Budhiraja to uniquely determine the priority order for loading the non-core services at run time responsive to a user interaction during each execution of the software application, so that individual users may load, as needed, non-core services that are not necessarily used by every user.

Hawkins in view of Budhiraja does not expressly disclose the limitation wherein a download of a first non-core service may be interrupted to begin download of a second non-core service responsive to the user interaction.

However, Raz discloses an analogous software engine for application loading a software application onto a user's machine (see, for example, the abstract). A download of a first sequence of modules may be interrupted to begin download of a second sequence of modules (see, for example, column 7, lines 5-18) responsive to user interaction (see, for example, column 5, line 63 to column 6, line 6). This enables the engine to dynamically adapt the order in which modules are downloaded as needed during execution of the software application (see, for example, column 4, lines 1-16).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the engine of Hawkins with the features taught by Raz such that a download of a first non-core service may be interrupted to begin download of a second non-core service responsive to the user interaction, so as to dynamically adapt the order in which non-core services are downloaded as needed during execution.

With respect to claim 2 (original), Hawkins also discloses the limitation wherein the engine is part of the core service and is loaded with the core service (see, for example, column 5, lines 57-67, which shows adding the engine to the first portion of the program, i.e. so that it will be loaded with the core services).

With respect to claim 3 (original), Hawkins also discloses the limitation wherein the engine commences operation upon completion of loading of the core service (see, for example, column 6, lines 4-8, which shows commencing operation of the engine immediately after the core services are loaded and instantiated).

With respect to claim 7 (previously presented), Hawkins also discloses the limitation wherein before loading the non-core services they are registered with the engine (see, for example, column 3, line 66 to column 4, line 9, which shows determining whether classes or services have been previously referenced; note that in order to make this determination, the classes or services are inherently registered with the engine when they are first instantiated).

With respect to claim 8 (previously presented), Hawkins also discloses the limitation wherein the engine checks a registration list of non-core services before loading a requested non-core service (see, for example, column 3, line 66 to column 4, line 9, which shows determining whether classes or services have been previously referenced, i.e. checking a registration list, prior to loading them).

With respect to claim 16 (original), Hawkins also discloses the limitation wherein the loading is downloading over the Internet (see, for example, column 3, lines 9-21, which shows downloading an Internet application).

With respect to claim 17 (currently amended), the limitations recited in the claim are analogous to those of claim 1 (see the rejection of claim 1 above).

With respect to claim 18 (original), the limitations recited in the claim are analogous to those of claim 2 (see the rejection of claim 2 above).

With respect to claim 19 (original), the limitations recited in the claim are analogous to those of claim 3 (see the rejection of claim 3 above).

With respect to claim 20 (original), the limitations recited in the claim are analogous to those of claim 7 (see the rejection of claim 7 above).

With respect to claim 22 (previously presented), Hawkins also discloses the limitation wherein upon interaction with the application by the user, the application requests the engine to load at least one of the non-core services (see, for example, column 4, lines 23-39, which shows loading non-core classes or services based on user interaction), and the engine checks a registration and gives the at least one non-core service top priority for loading (see, for example, column 3, line 66 to column 4, line 9, which shows determining whether classes or services have been previously referenced, i.e. checking the registration list; also see, for example, column 5, lines 57-67, which shows that the classes or services are loaded using a time-based priority order; note that top priority is inherently assigned to the class or service required first).

With respect to claim 32 (original), the limitations recited in the claim are analogous to those of claim 16 (see the rejection of claim 16 above).

With respect to claim 40 (previously presented), Hawkins also discloses the limitation wherein the application comprises a non-browser application (see, for example, column 6, lines 41-43, which shows that the application comprises a program executed in any client environment, such as a non-browser application).

6. Claims 9-15, 25-31 and 33-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hawkins in view of Budhiraja in view of Raz, as applied to claims 1 and 17 above, respectively, and further in view of U.S. Patent No. 6,430,570 to Judge et al. (art of record, "Judge 570").

With respect to claim 9 (original), although Hawkins discloses that class files may be cached on a client machine prior to execution (see, for example, column 1, lines 34-40), Hawkins does not expressly disclose the limitation wherein there is provided a cache into which at least one object for the application can be stored.

However, Judge 570 discloses the limitation above in terms of storing class objects in a cache to reduce the number of redundant downloads, thereby improving performance (see, for example, application cache 52 in FIG. 2 and column 7, lines 28-36).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the caching features taught by Judge 570 in the system of Hawkins, for the purpose of improving performance (see, for example, Judge 570, column 7, lines 28-29).

With respect to claim 10 (original), Hawkins does not expressly disclose the limitation wherein the engine includes a memory management module that keeps track of usage of cached objects; the memory management module being able to de-allocate one or more of the objects.

However, Judge 570 further discloses the limitation above in terms of an application memory manager that keeps track of cache objects and removes objects for garbage collection in order to free up space in memory (see, for example, column 8, lines 37-47).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the caching features taught by Judge 570 in the system of Hawkins, for the purpose of improving performance (see, for example, Judge 570, column 7, lines 28-29).

With respect to claim 11 (original), Hawkins also discloses the limitation wherein the cache is operative only when the application is on the user's machine (see, for example, column 1, lines 34-40, which shows that class files may be cached on a client machine prior to execution).

With respect to claim 12 (original), Hawkins does not expressly disclose the limitation wherein the cache includes an object repository into which the at least one object is placed, and an object description.

However, Judge 570 further discloses the limitation above in terms of caching class objects in a repository along with a description comprising an object reference (see, for example, application cache 52 in FIG. 2 and column 7, lines 45-51).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the caching features taught by Judge 570 in the system of Hawkins, for the purpose of improving performance (see, for example, Judge 570, column 7, lines 28-29).

With respect to claim 13 (original), Hawkins does not expressly disclose the limitation wherein the object description includes one or more selected from the group consisting of: object reference, object key, reference counter and time stamp.

However, Judge 570 further discloses the limitation above in terms of maintaining an object reference (see, for example, column 7, lines 45-51).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the caching features taught by Judge 570 in the system of Hawkins, for the purpose of improving performance (see, for example, Judge 570, column 7, lines 28-29).

With respect to claim 14 (original), Hawkins does not expressly disclose the limitation wherein the de-allocation of one or more of the objects includes an arbitrary time offset.

However, Judge 570 further discloses the limitation above in terms of continuously monitoring the free memory level and removing objects for de-allocation by the garbage collector as needed (see, for example, column 8, lines 43-47; note that continuously monitoring the free memory level inherently involves polling at an arbitrary time interval or offset).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the caching features taught by Judge 570 in the system of Hawkins, for the purpose of improving performance (see, for example, Judge 570, column 7, lines 28-29).

With respect to claim 15 (original), Hawkins does not expressly disclose the limitation wherein if the object description of an object in the object repository has a reference counter equal to zero for a time equal to at least the time offset, the corresponding object description is removed from the object repository.

However, Judge 570 further discloses the limitation above in terms of maintaining an object reference to each class to protect it from garbage collection (see, for example, column 7, lines 45-51), meaning that the objects will be removed when the reference counter is equal to zero. Judge 570 further discloses that the garbage collection will take place when a low or no free memory level is detected, inherently after an arbitrary time offset due to polling (see, for example, column 8, lines 43-47).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the caching features taught by Judge 570 in the system of Hawkins, for the purpose of improving performance (see, for example, Judge 570, column 7, lines 28-29).

With respect to claim 25 (original), the limitations recited in the claim are analogous to those of claim 9 (see the rejection of claim 9 above). Note that Judge 570 further discloses that the class objects are stored in the cache for later reuse (see, for example, column 7, lines 45-51).

With respect to claim 26 (original), the limitations recited in the claim are analogous to those of claim 10 (see the rejection of claim 10 above).

With respect to claim 27 (original), the limitations recited in the claim are analogous to those of claim 11 (see the rejection of claim 11 above).

With respect to claim 28 (original), the limitations recited in the claim are analogous to those of claim 12 (see the rejection of claim 12 above).

With respect to claim 29 (original), the limitations recited in the claim are analogous to those of claim 13 (see the rejection of claim 13 above).

With respect to claim 30 (original), the limitations recited in the claim are analogous to those of claim 14 (see the rejection of claim 14 above).

With respect to claim 31 (original), the limitations recited in the claim are analogous to those of claim 15 (see the rejection of claim 15 above).

With respect to claim 33 (previously presented), although Hawkins discloses that class files may be cached on a client machine prior to execution (see, for example, column 1, lines 34-40), Hawkins does not expressly disclose a computer memory management system including a cache, and wherein objects of the application are storable in the cache for reuse.

However, Judge 570 discloses the limitation above in terms of an application memory manager (see, for example, column 8, lines 37-42) that includes a cache for storing class objects (see, for example, application cache 52 in FIG. 2 and column 7, lines 28-36), so that the objects may be reused (see, for example, column 7, lines 45-51).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the caching features taught by Judge 570 in the system of Hawkins, for the purpose of improving performance (see, for example, Judge 570, column 7, lines 28-29).

With respect to claim 34 (previously presented), the limitations recited in the claim are analogous to those of claim 11 (see the rejection of claim 11 above).

With respect to claim 35 (previously presented), the limitations recited in the claim are analogous to those of claim 12 (see the rejection of claim 12 above).

With respect to claim 36 (previously presented), the limitations recited in the claim are analogous to those of claim 13 (see the rejection of claim 13 above).

With respect to claim 37 (previously presented), the limitations recited in the claim are analogous to those of claim 10 (see the rejection of claim 10 above).

With respect to claim 38 (previously presented), the limitations recited in the claim are analogous to those of claim 14 (see the rejection of claim 14 above).

With respect to claim 39 (previously presented), the limitations recited in the claim are analogous to those of claim 15 (see the rejection of claim 15 above).

7. Claim 42 is rejected under 35 U.S.C. 103(a) as being unpatentable over Hawkins in view of Budhiraja in view of U.S. Patent No. 6,430,564 to Judge et al. (now made of record, "Judge 564").

With respect to claim 42 (new), Hawkins discloses a software application (see, for example, the abstract), comprising:

(a) at least one core service to enable the software application to operate on a computer (see, for example, FIG. 1 and column 4, lines 10-12, which shows core classes or services to enable the application to operate);

(b) a software engine, downloaded onto the computer with the at least one core service, for loading the software application onto the computer (see, for example, column 3, lines 22-26, which shows a software engine downloaded to the computer with the core classes or services for loading the application);

(c) a plurality of non-core services, each of the plurality of non-core services providing a functionality of the software application (see, for example, column 4, lines 23-39, which shows a plurality of non-core services providing functionality), wherein the non-core services of the software application are downloaded onto the computer according to a priority order determined by the software engine (see, for example, column 3, lines 22-26, which shows that the software engine downloads class files of the non-core services, and column 5, lines 57-67, which shows that the class files of the non-core services are downloaded in the correct order using a time-based priority order).

Although Hawkins discloses that the software engine determines the priority order for loading the non-core services at run time responsive to a user interaction during execution of the software application (see, for example, column 3, lines 57-65, which shows analyzing user interaction with the program at run-time, and column 4, lines 10-39, which shows determining the order in which classes or services are required for providing functionality corresponding to user interaction) and to an order with which the non-core services register with the core service (see, for example, column 3, lines 65-67, which shows that the non-core services are registered

with the core service in order using load points), Hawkins does not expressly disclose the limitation wherein the software engine uniquely determines the priority order for loading the non-core services at run time responsive to a user interaction during each execution of the software application.

However, Budhiraja discloses an analogous software application (see, for example, the abstract). A core service of the application is loaded onto the user's machine (see, for example, column 9, lines 16-27, which shows loading a core applet) to enable the application to commence to operate on the user's machine (see, for example, column 9, lines 27-32), and non-core services are subsequently loaded (see, for example, column 9, lines 32-35, which shows component applets). Budhiraja discloses that the component manager (i.e., the engine) uniquely determines which component applets (i.e., non-core services) to load at run time responsive to a user interaction during each execution of the software application (see, for example, column 9, lines 35-47). This enables individual users of the software application to load, as needed, components that are not necessarily used by every user (see, for example, column 9, lines 7-13).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the software engine of Hawkins with the features taught by Budhiraja to uniquely determine the priority order for loading the non-core services at run time responsive to a user interaction during each execution of the software application, so that individual users may load, as needed, non-core services that are not necessarily used by every user.

Hawkins in view of Budhiraja does not expressly disclose:

(d) an object repository for storing previously used data objects and associating the previously used data objects with a unique key; and

(e) wherein when execution of one of the non-core services requires the generation of one of the previously used data objects, the previously used data object is retrieved from the object repository rather than being generated by one of the plurality of non-core services using the unique key.

However, Judge 564 discloses an object repository for storing previously used data objects and associating the previously used data objects with a unique key (see, for example, data manager 48 and data objects 32 in FIG. 3, and column 5, lines 6-23). Judge 564 further discloses that previously used data objects are retrieved from the object repository using the unique key rather than being generated (see, for example, column 5, lines 48-65), so as to flexibly manage memory usage when resources are constrained (see, for example, column 8, lines 34-42).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the object repository features of Judge 564 into Hawkins, so as to flexibly manage memory usage when resources are constrained.

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

Charli C Das
CHAMELI C. DAS
PRIMARY EXAMINER
4/12/06